# 安装插件

在BuildConfig.groovy的plugins配置块中增加如下代码：

|  |
| --- |
| compile ":tianan-security:0.1-SNAPSHOT" |

安装完成后记得更新项目依赖。

## 增加UM配置

在Config.groovy中增加如下配置：

|  |
| --- |
| // Added by the Spring Security Core plugin:  grails.plugin.springsecurity.controllerAnnotations.staticRules = [  '/': ['permitAll'],  '/index': ['permitAll'],  '/index.gsp': ['permitAll'],  '/\*\*/js/\*\*': ['permitAll'],  '/services/\*\*': ['permitAll'],  '/\*\*/css/\*\*': ['permitAll'],  '/\*\*/images/\*\*': ['permitAll'],  '/\*\*/favicon.ico': ['permitAll']]  //UM 配置  um.system.code = "VEHCOM"  um.wsdl.location = "http://10.100.137.178:8002/um/services/AuthoritiesPortType?wsdl"  cas.serviceUrl = "http://10.100.137.178:8001"  application.home.url = "http://localhost:8080/demo"  environments {  production {  cas.serviceUrl = "http://10.100.137.178:8001"  um.wsdl.location = "http://10.100.137.178:8002/um/services/AuthoritiesPortType?wsdl"  application.home.url = "http://localhost:8080/demo"  }  development {  cas.serviceUrl = "http://10.100.137.178:8001"  um.wsdl.location = "http://10.100.137.178:8002/um/services/AuthoritiesPortType?wsdl"  application.home.url = "http://localhost:8080/demo"  }  test {  cas.serviceUrl = "http://10.100.137.178:8001"  um.wsdl.location = "http://10.100.137.178:8002/um/services/AuthoritiesPortType?wsdl"  application.home.url = "http://localhost:8080/demo"  }  }  //cas  grails.plugin.springsecurity.cas.loginUri = '/login'  grails.plugin.springsecurity.cas.serviceUrl = "${application.home.url}*/j\_spring\_cas\_security\_check"*  grails.plugin.springsecurity.cas.serverUrlPrefix = "${cas.serviceUrl}"  grails.plugin.springsecurity.cas.proxyCallbackUrl = "${application.home.url}*/secure/receptor"*  grails.plugin.springsecurity.cas.proxyReceptorUrl = '/secure/receptor'  grails.plugin.springsecurity.logout.afterLogoutUrl = "${cas.serviceUrl}*/logout?url=$*{application.home.url}" |

## 验证插件是否生效

启动并访问应用，如果出现UM登陆界面，说明插件已经成功生效：



# 使用插件

## 使用annotation对url进行安全配置

插件默认使用annotation对url进行保护。使用的是@Secured（grails.plugin.springsecurity.annotation）。@Secured用于知名controller的action可以被拥有什么角色的用户访问。

可以对controller class做注解，这就说明这个controller的所有action都只能被拥有指名的角色的用户访问。也可以对controller的action进行注解。如果controller和action都有注解，则以action的注解为准。

例如下面的controller：

|  |
| --- |
| package com.mycompany.myapp  import grails.plugin.springsecurity.annotation.Secured  class SecureAnnotatedController {  @Secured(['ROLE\_ADMIN'])  def index() {  render 'you have ROLE\_ADMIN'  }  @Secured(['ROLE\_ADMIN', 'ROLE\_SUPERUSER'])  def adminEither() {  render 'you have ROLE\_ADMIN or SUPERUSER'  }  def anybody() {  render 'anyone can see this'  }  } |

上面的例子说明只有登陆用户且拥有ROLE\_ADMIN角色的用户可以访问myapp/secureAnnotated（或/myapp/secureAnnotated/index）。

而只有登陆用户且拥有ROLE\_ADMIN或者ROLE\_SUPERUSER角色的用户可以访问/myapp/secureAnnotated/adminEither。

任何用户都可以访问/myapp/secureAnnotated/anybody。

下面的例子对整个controller做注解：

|  |
| --- |
| package com.mycompany.myapp  import grails.plugin.springsecurity.annotation.Secured  @Secured(['ROLE\_ADMIN'])  class SecureClassAnnotatedController {  def index() {  render 'index: you have ROLE\_ADMIN'  }  def otherAction() {  render 'otherAction: you have ROLE\_ADMIN'  }  @Secured(['ROLE\_SUPERUSER'])  def super() {  render 'super: you have ROLE\_SUPERUSER'  }  } |

这样角色为ROLE\_ADMIN的用户可以访问myapp/secureClassAnnotated（或/myapp/ secureClassAnnotated /index）。

只有角色为ROLE\_SUPERUSER的用户可以访问/myapp/secureClassAnnotated/super。

除了具体的角色外，@Secured还可以使用IS\_AUTHENTICATED\_ANONYMOUSLY, IS\_AUTHENTICATED\_REMEMBERED, 和 IS\_AUTHENTICATED\_FULLY这样的组合角色token。

* IS\_AUTHENTICATED\_ANONYMOUSLY
  + signifies that anyone can access this URL. By default the AnonymousAuthenticationFilter ensures an 'anonymous' Authentication with no roles so that every user has an authentication. The token accepts any authentication, even anonymous.
* IS\_AUTHENTICATED\_REMEMBERED
  + requires the user to be authenticated through a remember-me cookie or an explicit login.
* IS\_AUTHENTICATED\_FULLY
  + requires the user to be fully authenticated with an explicit login.

## controllerAnnotations.staticRules

除了annotation以外，还可以在Config.groovy中增加staticRules配置，用来定义那些不会调用到controller的url的权限：

|  |
| --- |
| grails.plugin.springsecurity.controllerAnnotations.staticRules = [  '/': ['permitAll'],  '/index': ['permitAll'],  '/index.gsp': ['permitAll'],  '/\*\*/js/\*\*': ['permitAll'],  '/\*\*/css/\*\*': ['permitAll'],  '/\*\*/images/\*\*': ['permitAll'],  '/js/admin/\*\*': ['ROLE\_ADMIN'],  '/\*\*/favicon.ico': ['permitAll']] |

## 帮助类

spring security插件提供了不少帮助类，在代码和gsp中使用这些帮助类可以避免直接使用spring security的底层代码。

### Security TagLib

security插件提供的gsp标签可以用于基于用户权限显示页面内容和判断用户是否有请求某个action。标签使用sec namespace。

#### ifLoggedIn

如果当前是登陆用户则显示body里的内容：

|  |
| --- |
| <sec:ifLoggedIn>  Welcome Back!  </sec:ifLoggedIn> |

#### ifNotLoggedIn

如果是当前非登陆用户则显示body里的内容：

|  |
| --- |
| <sec:ifNotLoggedIn>  <g:link controller='login' action='auth'>Login</g:link>  </sec:ifNotLoggedIn> |

#### ifAllGranted

如果当前用户拥有标签列出的所有角色，则显示body的内容：

|  |
| --- |
| <sec:ifAllGranted roles="ROLE\_ADMIN,ROLE\_SUPERVISOR">secure stuff here</sec:ifAllGranted> |

#### ifAnyGranted

只要当前用户拥有的角色在标签列出的所有角色之中，则显示body的内容：

|  |
| --- |
| <sec:ifAnyGranted roles="ROLE\_ADMIN,ROLE\_SUPERVISOR">secure stuff here</sec:ifAnyGranted> |

#### ifNotGranted

如果当前用户的角色不在标签列出的角色之中，则显示body内容：

|  |
| --- |
| <sec:ifNotGranted roles="ROLE\_USER">non-user stuff here</sec:ifNotGranted> |

#### loggedInUserInfo

显示当前认证用户的信息，标签属性就是是代表当前用户的user类的属性：

|  |
| --- |
| <sec:loggedInUserInfo field="username"/> |
| Welcome Back <sec:loggedInUserInfo field="fullName"/> |
| <sec:loggedInUserInfo field="roleInfos"/> |

#### username

显示当前认证用户的用户名：

|  |
| --- |
| <sec:ifLoggedIn>  Welcome Back <sec:username/>!  </sec:ifLoggedIn>  <sec:ifNotLoggedIn>  <g:link controller='login' action='auth'>Login</g:link>  </sec:ifNotLoggedIn> |

#### ifSwitched

如果当前用户是从另一个用户切换（switch）而来，则显示body内容：

|  |
| --- |
| <sec:ifLoggedIn>  Logged in as <sec:username/>  </sec:ifLoggedIn>  <sec:ifSwitched>  <a href='${request.contextPath}/j\_spring\_security\_exit\_user'>  Resume as <sec:switchedUserOriginalUsername/>  </a>  </sec:ifSwitched>  <sec:ifNotSwitched>  <sec:ifAllGranted roles='ROLE\_SWITCH\_USER'>  <form action='${request.contextPath}/j\_spring\_security\_switch\_user' method='POST'>  Switch to user: <input type='text' name='j\_username'/><br/>  <input type='submit' value='Switch'/> </form>  </sec:ifAllGranted>  </sec:ifNotSwitched> |

#### ifNotSwitched

如果当前用户不是从另一个用户switch而来，则显示body内容，例子同上。

#### switchedUserOriginalUsername

如果当前用户是从另一个用户switch而来，显示原先那个用户的用户名，例子同上。

#### access

当标签中的布尔表达式返回值为true时或者url属性中的url有权限访问时，显示body内容：

|  |
| --- |
| <sec:access expression="hasRole('ROLE\_USER')">  You're a user  </sec:access> |
| <sec:access url="/admin/user">  <g:link controller='admin' action='user'>Manage Users</g:link>  </sec:access> |

标签也可以使用controller和action属性，当用户有权限访问controller的action时，显示body内容：

|  |
| --- |
| <sec:access controller='admin' action='user'>  <g:link controller='admin' action='user'>Manage Users</g:link>  </sec:access> |

或者使用命名的url mapping：

|  |
| --- |
| <sec:access mapping='manageUsers'>  <g:link mapping='manageUsers'>Manage Users</g:link>  </sec:access> |

甚至可以使用createLink：

|  |
| --- |
| <sec:access url='${createLink(controller: 'admin', action: 'user', base: "/")}'>  <g:link controller='admin' action='user'>Manage Users</g:link>  </sec:access> |

注意使用base: "/"，以避免在生成的url中出现应用的context name。

#### noAccess

noAccess的用户正好和access标签相反。

### SpringSecurityService

grails.plugin.springsecurity.SpringSecurityService类提供一些和安全相关的工具方法。可以使用如下方法把这个service注入到controller、service等grails的artefact类型中：

|  |
| --- |
| def springSecurityService |

#### getCurrentUser()

获取代表当前登陆用户的domain对象:

|  |
| --- |
| class SomeController {  def springSecurityService  def someAction() {  def user = springSecurityService.currentUser  …  }  } |

#### isLoggedIn()

判断当前用户是否是认证用户：

|  |
| --- |
| class SomeController {  def springSecurityService  def someAction() {  if (springSecurityService.isLoggedIn()) {  …  }  else {  …  }  }  } |

#### getAuthentication()

获取当前用户的[Authentication](http://docs.spring.io/spring-security/site/docs/3.2.x/apidocs/org/springframework/security/core/Authentication.html)，如果是登陆用户则就是一个[UsernamePasswordAuthenticationToken](http://docs.spring.io/spring-security/site/docs/3.2.x/apidocs/org/springframework/security/authentication/UsernamePasswordAuthenticationToken.html)对象。如果是非登陆用户，则是一个grails.plugin.springsecurity.authentication. GrailsAnonymousAuthenticationToken，这个对象用org.springframework.security.core.userdetails.User对象作为其Principal，并有一个ROLE\_ANONYMOUS的角色。例如：

|  |
| --- |
| class SomeController {  def springSecurityService  def someAction() {  def auth = springSecurityService.authentication  String username = auth.username  def authorities = auth.authorities // a Collection of GrantedAuthority  boolean authenticated = auth.authenticated  …  }  } |

#### getPrincipal()

获取当前用户的Principal，对于登陆用户来说得到的principal是grails.plugin.springsecurity.userdetails.GrailsUser对象，如果使用的是自定义的UserDetailsService，则是一个自定义的[UserDetails](http://docs.spring.io/spring-security/site/docs/3.2.x/apidocs/org/springframework/security/core/userdetails/UserDetails.html)对象。

如果是非登陆用户且启用了[AnonymousAuthenticationFilter](http://docs.spring.io/spring-security/site/docs/3.2.x/apidocs/org/springframework/security/web/authentication/AnonymousAuthenticationFilter.html)，则principal是一个标准的org.springframework.security.core.userdetails.User对象。

使用示例如下：

|  |
| --- |
| class SomeController {  def springSecurityService  def someAction() {  def principal = springSecurityService.principal  String username = principal.username  def authorities = principal.authorities // a Collection of GrantedAuthority  boolean enabled = principal.enabled  …  }  } |

## 国际化

在properties文件中增加如下信息：

|  |
| --- |
| error.403.title = 403 - 拒绝访问的页面!  error.403.message = 对不起，您没有权限访问这个页面!  error.403.callout = 出错啦!  error.button.backToHome = 返回首页  error.button.contactSupport = 联系客服 |

## 拒绝访问页面

拒绝访问的页面为:grails-app/views/login/denied.gsp，可对该页面进行定制。

## 登出

如果需要在用户退出后从cas服务器上单点登出，可以在Config.groovy中增加如下配置:

|  |
| --- |
| grails.plugin.springsecurity.logout.afterLogoutUrl = "${cas.serviceUrl}/logout?url=${application.home.url}" |

应用中用于推出的uri可以如下产生：

|  |
| --- |
| <a href=*"*${createLink(uri: '/j\_spring\_security\_logout')}*"*> <i  class=*"icon-off"*></i> 退出  </a> |